Day 7: Handy Haversacks

(Povezava na nalogo)

Na letališču imajo neka pravila v zvezi s tem, kakšne torbe morajo biti v kakšnih torbah.

```
light red bags contain 1 bright white bag, 2 muted yellow bags.
dark orange bags contain 3 bright white bags, 4 muted yellow bags.
bright white bags contain 1 shiny gold bag.
muted yellow bags contain 2 shiny gold bags, 9 faded blue bags.
shiny gold bags contain 1 dark olive bag, 2 vibrant plum bags.
dark olive bags contain 3 faded blue bags, 4 dotted black bags.
vibrant plum bags contain 5 faded blue bags, 6 dotted black bags.
faded blue bags contain no other bags.
dotted black bags contain no other bags.
```

Če izpustimo zgodbico, sta vprašanji preprosto:

1. Pri koliko različnih začetnih torbah bomo nekoč naleteli na zlato torbo?
2. Če začnemo z zlato torbo, koliko torb bo vsebovala (vključno s torbami, vsebovanimi v torbah ...)

Branje podatkov

Advent of Code 2019 se je sukal okrog programiranja nekega navideznega stroja, poln je bil drobnih algoritmičnih cukrčkov. Letošnji bo žal očitno na temo branja podatkov. Veliko užitkov želim vsem, ki ste se ga odločili delati v čistem C-ju.

Shranimo eno od vrstic v niz in poglejmo, kako jo zmrcvariti v ustrezne dele.

```
line = "light red bags contain 1 bright white bag, 2 muted yellow bags."
```

Dan bo rešila metoda nizov `partition(x)`. Kot argument dobi podniz `x` in vrne del originalnega niza pred `x`, `x` in del za `x`. Torej:

```
outer, _, inners = line.partition(" bags contain ")
```

```
outer
```

```
'light red'
```

```
inners
```

```
'1 bright white bag, 2 muted yellow bags.'
```

`inner` zdaj razsekamo glede na vejico, podnize pa na presledke. Od podnizov vzamemo prvi kos, in ostale, zadnjega pa zavržemo v `_`.

```
for inner in inners.split(", "):
    n, *color, _ = inners.split()
    print(n, color)
```

```
1 ['bright', 'white', 'bag,', '2', 'muted', 'yellow']
1 ['bright', 'white', 'bag,', '2', 'muted', 'yellow']
```

Kdor ne razume tistega z zvezdico, naj se spomni razpakiranja terk.

```
a, b, c, *d, e, f = (1, 2, 3, 4, 5, 6, 7, 8, 9)
```

```
c
```

```
3
```

```
d
```

```
[4, 5, 6, 7]
```

```
e
```

```
8
```

```
f
```

```
9
```

`n` je potrebno pretvoriti v `int`, `color` pa z `" ".join` združiti nazaj v en niz.

Zložimo vse skupaj, pa imamo branje.

```
rules = {}
for line in open("example.txt"):
    outer, _, inners = line.partition(" bags contain ")
    inn_bags = {}
    for inner in inners.split(","):
        n, *color, _ = inner.split()
        if n != "no":
            inn_bags[" ".join(color)] = int(n)
    rules[outer] = inn_bags
```

Za `if n != "no"` smo poskrbeli za torbe, ki ne vsebujejo drugih torb. Kaj je v tem primeru v `color` in `_` nam je očitno popolnoma vseeno.

```
rules
```

```
{'light red': {'bright white': 1, 'muted yellow': 2},
 'dark orange': {'bright white': 3, 'muted yellow': 4},
 'bright white': {'shiny gold': 1},
 'muted yellow': {'shiny gold': 2, 'faded blue': 9},
 'shiny gold': {'dark olive': 1, 'vibrant plum': 2},
 'dark olive': {'faded blue': 3, 'dotted black': 4},
 'vibrant plum': {'faded blue': 5, 'dotted black': 6},
 'faded blue': {},
 'dotted black': {}}
```

Izbrana oblika - slovar slovarjev bo najbolj praktična za obe funkciji, ki ju moramo napisati.

Branje z regularnimi izrazi

Z regularnim izrazom je praktično zgrabiti posamično notranjo torbo. Izraz, ki ga potrebujemo, je `(\d+?) (.+?) bags?[,.]`. Regularni izrazi imajo metodo `findall`, ki vrne skupine v vseh ponovitvah regularnega izraza, takole:

```
import re
re_content = re.compile(r"(\d+?) (.+?) bags?[,.]")

inners = "1 bright white bag, 2 muted yellow bags."

re_content.findall(inners)
```

```
[('1', 'bright white'), ('2', 'muted yellow')]
```

Podatke tako preberemo z

```
rules = {outer: {g[1]: int(g[0]) for g in re_content.findall(inners)}
         for outer, _, inners in (line.partition(" bags contain ") for line in open("input.t
```

Izraz preberimo od zadaj (kot vedno v Pythonu - Kotlin je tu lepši prav zato, ker gredo stvari naprej, ne nazaj!).

Najprej imamo `line.partition(" bags contain ") for line in open("input.txt")`, ki gre čez datoteko in vsako vrstico prelomi okrog `" bags contain "`. Čez ta generator spustimo zanko `for outer, _, inners`.

To je bila druga vrstica. V prvi pa iz teh `outer` in `inners` sestavimo ključe in vrednosti slovarja. Ključ je očitno `outer`. V `inner` moramo najti vse pojavitve vzorca, ki ga opisuje regularni izraz, `for g in re_content.findall(inners)`. Barvo (`g[0]`) bomo uporabili kot ključ, število (`int(g[0])`) bo vrednost.

Rešitev

Najbolj klasična, najbolj začetniška rekurzija.

Ni kaj komentirati.

```
def contains(t, x):
    return t == x or any(contains(y, x) for y in rules[t])

def count(t):
    return 1 + sum(n * count(x) for x, n in rules[t].items())

print(sum(contains(bag, "shiny gold") for bag in rules) - 1)

print(count("shiny gold") - 1)
```

```
233
421550
```